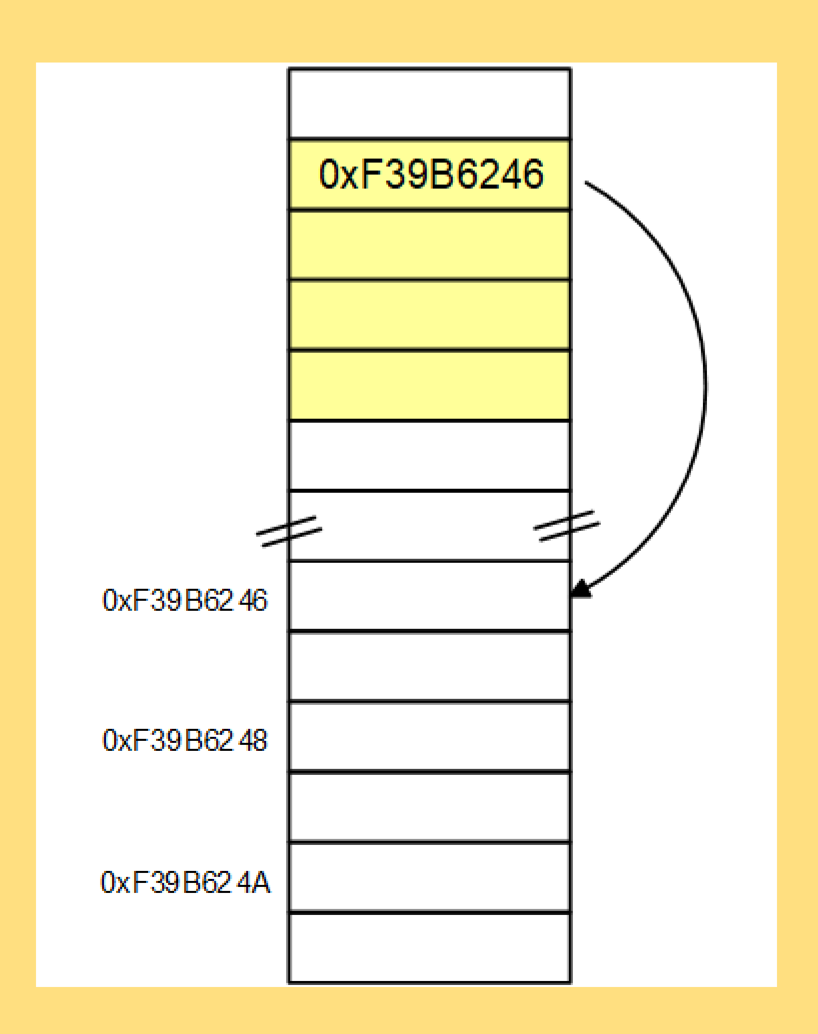
PUNTEROS

Un puntero es un tipo de dato que permite almacenar una dirección de memoria.

Esto quiere decir que un puntero apunta a una determinada ubicación en la memoria.



DECLARACIÓN

Para poder declarar una variable de tipo puntero se necesita asignarle un nombre precedido por un tipo asociado y el operador de indirección (*).

Donde tipo asociado es cualquier tipo de dato, tanto los definidos por el compilador, como los definidos por el usuario.

SINTAXIS:

tipo asociado * nombre puntero;

```
int * punt;
```

punt es un puntero a entero.

TAMAÑO DE LOS PUNTEROS

Como un puntero puede almacenar una dirección de memoria su tamaño será el adecuado para poder guardar en forma adecuada dicho valor.

Este valor es dependiente del sistema operativo para sistemas operativos de 32 bits su tamaño será de 4 bytes.

OPERADORES

Los punteros tienen asociados dos operadores, el operador de direccionamiento (&) y el operador de indireccionamiento (*).

OPERADOR DE DIRECCIONAMIENTO (&)

Este operador permite poder obtener la dirección de una variable.

```
int a;
int *pi;
pi=&a;
```

OPERADOR DE INDIRECCIÓN (*)

Este operador permite acceder al contenido de la memoria que es apuntada por el puntero.

```
int a,*pi;
pi=&a;
*pi=4;
```

El significado de estas líneas es que como el puntero pi apunta a la variable a, al hacer que mediante el puntero acceda al contenido de la memoria se escribió en esa dirección el valor 4, modificando indirectamente el valor de la variable a.

Ejercitación:

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt01.c

OPERACIONES QUE ADMITEN LOS PUNTEROS

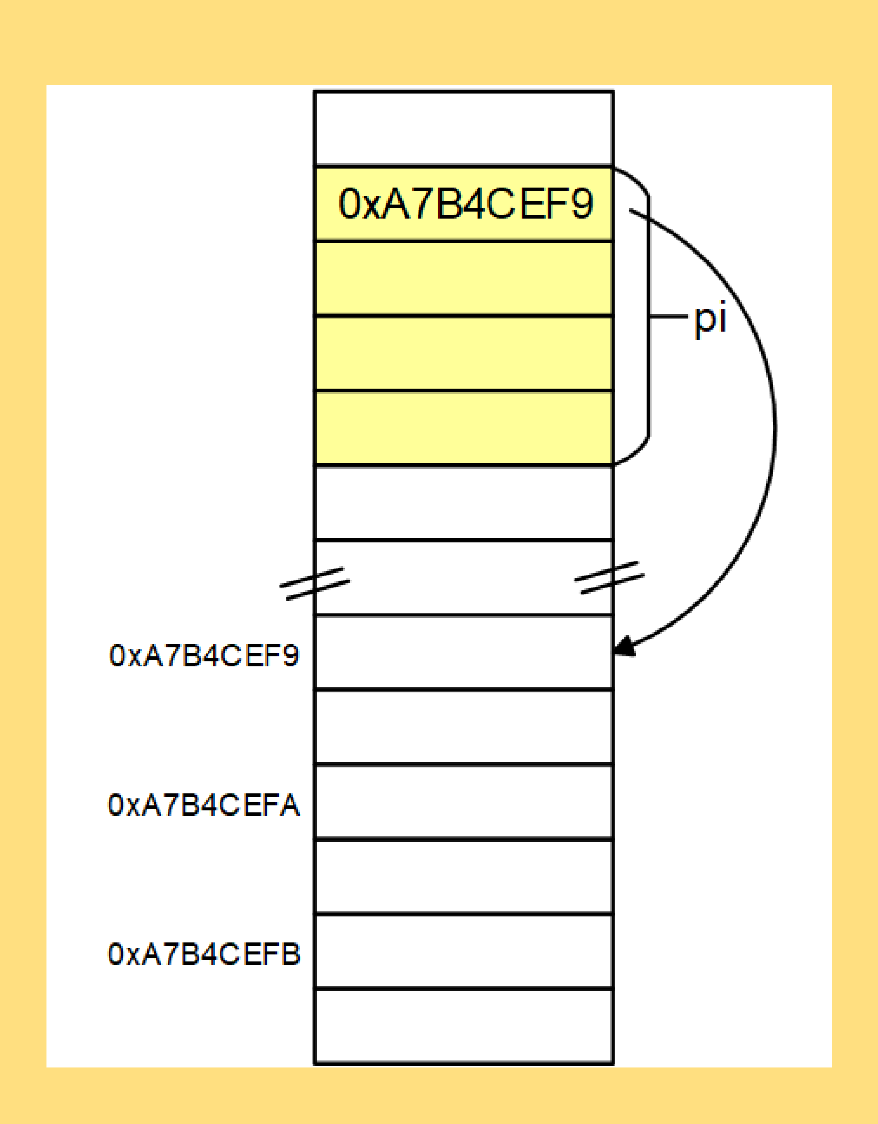
ASIGNACIÓN (=)

A un puntero se le puede asignar un valor entero, el significado es una dirección de memoria.

Debemos recordar que la memoria está dividida en celdas de 1 byte cada una, por lo que entre una posición y la siguiente la distancia que los separa es uno (1).

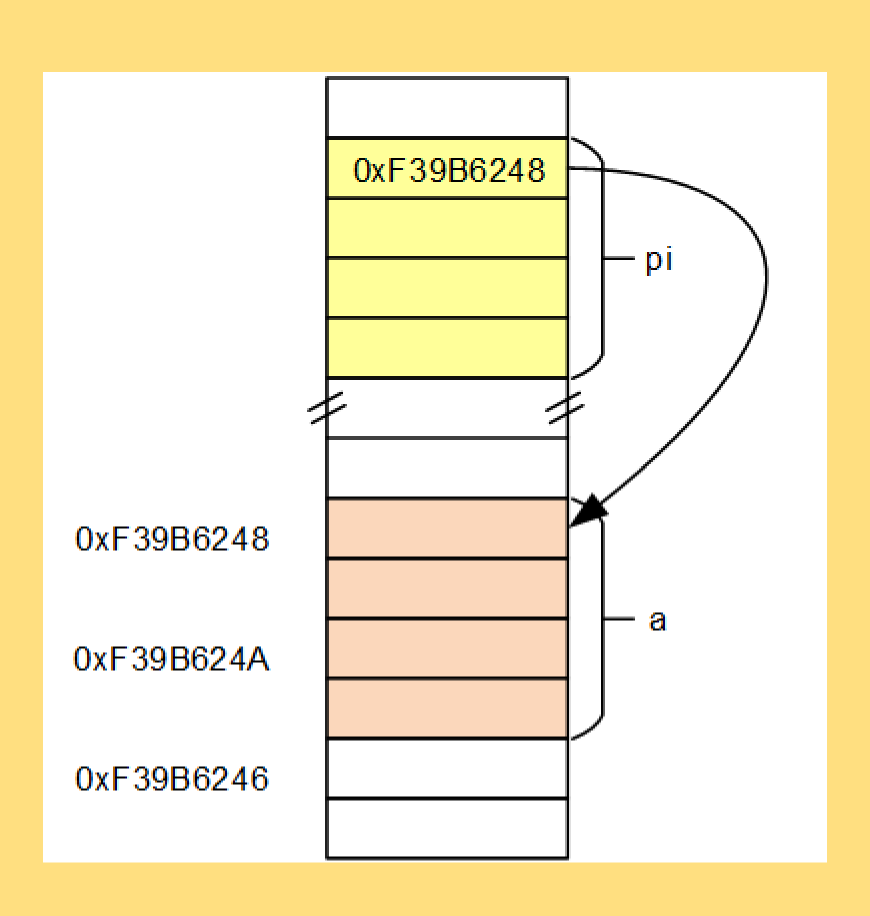
El operador que se utiliza es el de asignación (=).

```
int *pi;
pi=0xA7B4CEF9;
```



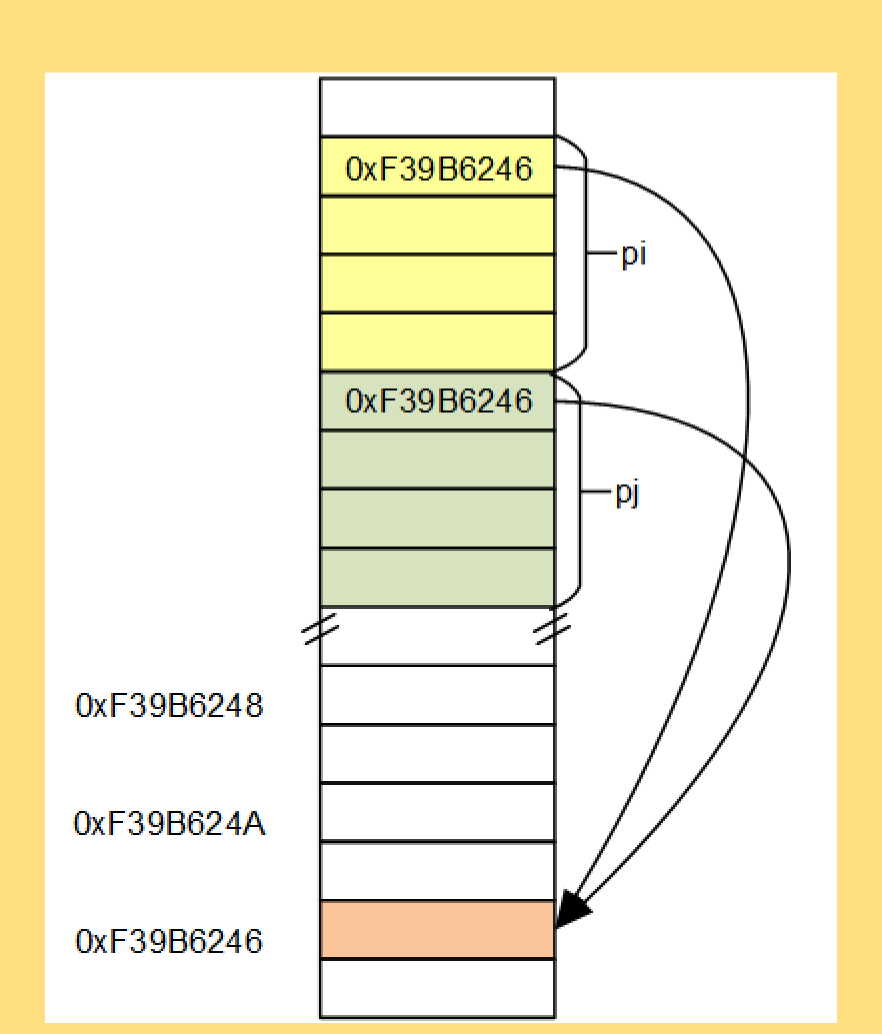
También admiten la asignación de la dirección de una variable, para realizar esta operación se utiliza el operador de direccionamiento (&).

```
int a;
pi=&a;
```



Otro valor que se le puede asignar a un puntero es otro puntero, como es general en C a toda variable de un determinado tipo se le puede asignar otra variable del mismo tipo.

```
int *pj,*pi;
pj=0xF39B6246;
pi=pj;
```



Hay que tener en cuenta que cuando se declara un puntero hay que inicializarlo, ya que el valor que contiene luego de la declaración es basura, y en este caso se dice que el puntero esta descontrolado y puede ocurrir que se cometan errores muy peligrosos en el caso de utilizarlo.

<u>Ejercitación:</u>

Utilice el compilador , ejecuta paso a paso los siguientes programas

***Punt02.c, Punt03.c, Punt04.c*.**

ARITMÉTICAS

Admiten las operaciones de suma y resta.

Se utilizan los operadores de adición (+) y sustracción (-).

Los valores que se le pueden adicionar o sustraer pueden ser enteros u otro puntero.

Si se le adiciona un entero el resultado es que se va a desplazar hacia posiciones superiores de la memoria, en cambio si se le sustrae un entero este se desplazará hacia posiciones inferiores.

La cantidad de bytes que se desplaza el puntero es el tamaño del tipo asociado

Otro dato que se puede restar a un puntero es el de otro puntero

El resultado será el de un entero

Significa la distancia en tipos asociados que separan a ambos punteros.

Este valor puede ser positivo, que indica que el puntero que esta como minuendo está apuntando a una dirección más alta de la memoria, si fuese negativo indicaría lo contrario.

Hay que tener mucho cuidado con esta interpretación ya que no indica la distancia expresada en bytes.

La operación de suma de punteros no es posible realizarla ya que carece de significado.

Las operaciones de producto o cociente no son aceptadas por los punteros.

Otra imposibilidad es la de realizar operaciones utilizando datos de tipo flotante.

Las operaciones de desplazamiento o las operaciones lógicas no se pueden efectuar con los punteros.

Ejercitación:

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt05.c, Punt06.c, Punt07.c y Punt08.c**.**

COMPARACIÓN

Los punteros admiten las operaciones de comparación (<,>,==,!=,<=,>=)

Si un puntero es mayor que el otro esto indica que el mayor apunta a una dirección de memoria más alta que el otro

Si fueran iguales ambos apuntarían a la misma dirección

Si fuesen distintos ambos apuntarían a diferentes posiciones de memoria.

Que la comparación puede ser efectuada entre dos punteros, un puntero y un entero o entre un puntero y la dirección de una variable.

Ejercitación:

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt09.c **y** ***Punt10.c*.**

PUNTEROS Y VECTORES

Si se declara un vector, el compilador reservara la cantidad de memoria necesaria para el mismo, comenzando desde la posición cero (0).

Cuando se hace referencia a un elemento determinado del vector, se utiliza el nombre seguido de un subíndice.

El compilador asigna al nombre del vector la dirección de comienzo de este, y el subíndice lo utiliza para desplazarse.

Si se utiliza un puntero para apuntar al comienzo del vector, se puede obtener el mismo resultado que utilizando el nombre del vector subindicado. Pero para poder realizar esto se debe utilizar la aritmética de punteros.

Como el nombre del vector contiene su dirección de comienzo, en realidad es un puntero constante, por lo que le podemos aplicar la aritmética de punteros y lograríamos el mismo efecto.

Si se utiliza un puntero apuntando a la dirección de comienzo de un vector

Se puede utilizar un puntero con un subíndice se obtendría el mismo resultado que subindicando a un vector

Esto se denomina dualidad puntero – vector.

Ejercitación:

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt11.c, Punt12.c, Punt13.c **y** ***Punt14.c*****.**

PASAJE DE PARÁMETROS POR REFERENCIA EN UNA FUNCIÓN

Cuando se pasa un parámetro a una función se lo hace mediante una copia del valor que se le envía, a este método se lo denomina pasaje por valor.

El parámetro que recibe la función es copiado en una variable que se declara en la misma

Como es una copia cualquier modificación que se realice en esta variable no tendrá efecto sobre el valor original.

Si se desea que cualquier modificación sobre el contenido de la copia se vea reflejado en la variable original se debe utilizar lo que se denomina pasaje por referencia.

El pasaje de parámetros por referencia en C se realiza utilizando punteros

El puntero se copia la dirección de la variable que se quiere modificar y en la función se utiliza el operador de indirección para acceder a su contenido.

Esto hace que el valor que se modifique en la función aparecerá modificado en la variable original.

Si se modifica el contenido del puntero, este apuntara a otra variable produciendo un resultado no esperado.

<u>Ejercitación:</u>

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt15.c **y** ***Punt16.c*.**

UNTEROS A STRING

Un puntero a string no es más que un puntero a char

Un string no es nada más que un vector de char

Tiene la característica de que está finalizado por el carácter “null”.
Son aplicables los mismos conceptos que para cualquier otro vector.
Hay que tener en cuenta una particularidad de los punteros a string

Cuando se declara un puntero a char también se lo inicializa, se lo puede hacer de dos formas distintas:

Asignarle la dirección de una variable, en este caso será como cualquier inicialización de un puntero.

Inicializarlo con un string, que será tomado como una constante.

El compilador reserva el espacio de memoria necesario para el string y lo ubica en esa posición, y le asigna al puntero la dirección de memoria correspondiente.

Hay que tener cuidado en no modificar el contenido del puntero, ya que como el compilador reservo el espacio suficiente y no conocemos la dirección en donde esto ocurrió, al modificar al puntero perderemos la localización del string y no podrá ser vuelto a usar.

Hay que tener cuidado en no exceder el largo del string utilizado en la inicialización ya que solo está reservado el lugar necesario para este y si nos excedemos podemos ocupar un lugar destinado para otro valor y esto podría traer inconvenientes en la ejecución del programa.

PUNTERO A ESTRUCTURA

Una estructura es un tipo de dato definido por el usuario, y si después de definirlo se utiliza como tipo asociado de un puntero esto es totalmente valido.

La estructura como sabemos es un tipo de dato que contiene campos, distintos tipos válidos, y se puede acceder a ellos mediante el operador de acceso a miembro (.).

Esto es válido cuando se utilizan punteros ya que si un puntero apunta a una variable de tipo estructura a través del operador de indirección se puede acceder a su contenido, pero como el contenido de esta son campos, se le debería agregar el operador de acceso a miembro.

Esto trae aparejado un inconveniente debido a la precedencia de los operadores, el operador de acceso a miembro tiene precedencia sobre el de indirección, por lo que habría que encerrar entre paréntesis al operador de indirección y al puntero para poder acceder a los campos de la estructura mediante el uso del operador de acceso a miembro.

Habría que tener en cuenta la precedencia de los operadores

Para solucionar este inconveniente se utiliza un nuevo operador, que se denomina operador apuntador de estructura (->).

Este operador permite el acceso directo a los campos mediante la utilización de un puntero.

PUNTERO A PUNTERO

Un puntero a puntero es solo un puntero que apunta a otro puntero.

Para declararlo deberemos utilizar el operador de indirección dos veces y precedido por un tipo asociado.

po asociado ** nombre;

```
float **pf;
```

pf es un puntero a puntero
a float

El tamaño de un puntero a puntero es el mismo que el de un puntero, ya que va a almacenar una dirección de memoria.

<u>Ejercitación:</u>

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt17.c **y** ***Punt18.c*****.**

VECTOR DE PUNTEROS

Un vector de punteros no es otra cosa que un vector cuyo tipo son punteros.

Como todo vector su nombre es un puntero a la dirección de comienzo, pero en este caso como los elementos del vector son punteros, este es un puntero a puntero.

DECLARACIÓN

tipo asociado * nombre [tamaño];

```
int * vec[10];
```

vec es un vector de punteros a entero.

Ejercitación:

Utilice el compilador , ejecuta paso a paso los siguientes programas

Punt19.c